COBB *Tips & techniques for MS-DOS and PC-DOS*

September 1990
Vol. 1 No. 5

Inside DOS™

Soliciting input in a batch file

The Debug script for KEYPRESS.COM was submitted by Marco Mason. Marco is Editor-in-Chief of two Cobb Group developer journals: Inside Microsoft C *and* Inside Turbo C.

If you write your own batch files, you'll sometimes want to solicit some input from the user, then execute a specific set of batch instructions depending on which key he or she presses. Although DOS doesn't provide a built-in command for this purpose, you can use Debug to create a program called KEYPRESS.COM that performs this function for you.

In this article, we'll show you how to use Debug to create KEYPRESS.COM. (As you'll see, KEYPRESS.COM can check to see whether the user pressed Y, N, or the ESC key.) We'll also show you how to use the result of KEYPRESS.COM along with an IF command to branch to a particular place in a batch file. Once you understand the basic testing and branching technique, you should read the article on page 8, which shows you how to use this technique in a more complex batch file.

Figure A

```
e100
b4 08 cd 21 24 df 3c 59 75 04 b0 03 eb 0e 3c 4e
e110
75 04 b0 02 eb 06 3c 1b 75 06 b0 01 b4 4c cd 21
e120
b2 07 b4 02 cd 21 eb d8
n keypress.com
rcx
28
w
q
```

To create the utility KEYPRESS.COM, first use a text editor to enter these instructions into the file KEYPRESS.SCR.

Creating KEYPRESS.COM

As we mentioned, you'll need to create a utility called KEYPRESS.COM to solicit user input in a batch file. Fortunately, you can create a copy of KEYPRESS.COM using DOS' built in byte editor—Debug.

As you may recall from our discussion of Debug last month, you should use a text editor (like Edlin) to create a script of Debug instructions, and then "feed" those instructions to Debug with a single command, rather than work with Debug interactively. Figure A shows a script of Debug instructions that will create the file KEYPRESS.COM.

If you are using Edlin, begin creating the Debug script by issuing the command

```
C>edlin keypress.scr
```

(Because EDLIN is an external DOS command, your computer's path must include your DOS directory or this command will return the message *Bad command or file name*.) At this point, DOS will respond

```
New file
*_
```

To begin inserting lines into the new file, type the letter *i* and press ↵:

```
*i
     1:*_
```

Now, enter the text shown in Figure A one line at a time. When you reach the end of a line, simply press ↵ to move on to the next line.

When you've typed last line in Figure A, type ↵ to bring up the prompt for line 12, then press [Ctrl]c. When you do this, Edlin will display the characters ^C

Continued on page 10

IN THIS ISSUE

COBB

Inside DOS™

Inside DOS (ISSN 1049-5320) is published monthly by The Cobb Group, Inc., 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220. Domestic subscriptions: 12 issues, $39. Foreign: 12 issues, $59. Single issues: $4 domestic, $6.50 foreign. Send subscriptions, fulfillment questions, and requests for bulk orders to Customer Relations, 9420 Bunsen Parkway, Suite 300, Louisville, KY 40220, or call toll free 1-800-223-8720. Address correspondence and special requests to The Editor, *Inside DOS*, at the address above.

POSTMASTER: Send address changes to The Cobb Group, Inc., P.O. Box 35160, Louisville, KY 40232. Second class postage is paid in Louisville, KY.

Editor-in-Chief:	Mark W. Crane
Contributing Editors:	Van Wolverton
	Chris DeVoney
	Tim Landgrave
Editing:	Jim Welp
	Toni Frank Bowers
Production:	Eric Paul
Design:	Karl Feige
Publications Director:	Katherine Chesky
Publisher:	Douglas Cobb

Conventions

To avoid confusion, we would like to explain a few of the conventions used in *Inside DOS*.

As you probably know, Microsoft has released several versions of DOS. Although most of the articles in this journal apply to all versions of DOS since 2.00, we'll sometimes point out a particular command or technique that requires a particular version. For simplicity, we'll refer to all 4.xx versions as version 4.

Typically, we'll use DOS's default prompt, C>, to represent the DOS prompt in our examples. If you have defined a custom DOS prompt, just assume that the C> represents your prompt.

When we instruct you to type something, those characters usually appear on a separate line along with the DOS prompt. The characters you type will appear in color, while the characters DOS displays will appear in black.

Occasionally, we won't display the command you type on a separate line. In these cases, we'll display the characters you type in italics. For example, we might say, "issue the command *dir *.txt* at the DOS prompt." Although DOS is not sensitive to capitalization, we'll always display the characters you type in lowercase.

When we refer to a general DOS command (not the command you actually type at the DOS prompt), we'll display that command name in all caps. For example, we might say, "You can use either the COPY or XCOPY command to transfer files from one disk to another."

Many commands accept parameters that specify a particular file, disk drive, or other option. When we show you the form of such a command, its parameters will appear in italics. For example, the form of the COPY command is:

```
copy file1 file2
```

where *file1* and *file2* represent the names of the source file and the target file, respectively.

The names of keys, such as [Shift], [Ctrl], and [F1], appear in brackets. The [Enter] key is represented by the symbol ↵. When two keys must be pressed simultaneously, those key names appear side by side, as in [Ctrl][Break] or [Ctrl]z.

More timesaving command line keys

Q I found your articles on DOS' command line editing keys in the May and June issues very helpful. In addition to those you presented, I've discovered a couple of other timesaving keys that really save time: the [Esc] key and the [F5] key. Here's how they work:

Suppose you've just typed a long command at the DOS prompt, and you're about to press ↵ to issue the command. But then you glance back across the command line and you realize you've mistyped one of the first few characters of the command.

You now can either press the [Backspace] key several times to back up and fix the mistake, or you can press ↵ to issue what you know is an invalid command, and retype the command again after DOS returns you to the DOS prompt.

Here's where the [Esc] key comes in handy. Pressing the [Esc] key tells DOS to cancel your current command and to let you retype the command from scratch. For example, suppose you want to copy the file C:\WORD\LETTER.DOC to drive A, and you type the command

```
C>copu c:\word\letter.doc a:_
```

Just before you press ↵, however, you realize that you've typed *copu* instead of *copy*. To correct the mistake, you can simply press [Esc], which will place a backslash at the end of the command line and will move the cursor below the letter *c* in copy:

```
C>copu c:\word\letter.doc a:\
  _
```

Although you don't see a new prompt at this point, you can retype the command just as you normally would, like this:

```
C>copu c:\word\letter.doc a:\
  copy c:\word\letter.doc a:_
```

Now that you've typed the command properly, just press ↵ to issue the corrected command.

Although the [Esc] key provides a nice way to exit the current command line, the above example really calls for another timesaving key: [F5]. When you press the [F5] key, DOS cancels the current command, and it copies the current command line to the command line buffer. Then, you can take advantage of the →, ←, and [F3] keys as you correct the command.

Returning to the previous example, suppose you've typed

```
C>copu c:\word\letter.doc a:_
```

and that you need to change the word *copu* to *copy*. If you press the [F5] key at this point, DOS will display

an At symbol (@) at the end of the current command line, and will move the cursor below the letter *c* in copy, like this:

```
C>copu c:\word\letter.doc a:@
  _
```

At this point, you can press the → key three times to copy the letters *cop* to the command line and to move the cursor below the letter *u*:

```
C>copu c:\word\letter.doc a:@
  cop_
```

Now you can type the letter *y*, and press the [F3] key to copy the remainder of the previous command line:

```
C>copu c:\word\letter.doc a:@
  copy c:\word\letter.doc a:_
```

Finally, just press ↵ to issue the corrected command.

I hope your readers will benefit from using the [Esc] and [F5] keys as much as I do.

Eddie Malphrus
Beaufort, South Carolina

A Mr. Malphrus' explanation of the [Esc] and [F5] keys nicely augments our discussion in the May and June issues about DOS' other command line editing keys. Once you've taken the time to understand how all of these keys work, you'll find yourself working much more efficiently at the DOS prompt.

DOS utility programs

Q Many of my cohorts have recently suggested that I check into purchasing a DOS utility package, such as Norton Utilities or PC Tools. Are these utility packages really all that useful, and if so, which one would you recommend I buy?

John Gavitt
Baltimore, Maryland

A Dozens of DOS utilities are currently available for a variety of prices. Picking the right one is not much different from picking the right word processor or spreadsheet application—each DOS utility has its own advantages and disadvantages. Although we can't recommend a specific DOS utility for you right away, we've asked Contributing Editor Van Wolverton to write an article on DOS utilities in a future issue of *Inside DOS*. In that article, Van will discuss the significant advantages that most DOS utilities have to offer, and he'll give you some guidelines for deciding which, if any, DOS utility package is right for you.

Cancelling a command

Q One of DOS' handiest features is the [Ctrl][Break] key combination. Whenever I issue a command, then decide I want to cancel it, I simply press [Ctrl][Break], and DOS will usually cease whatever it is doing at the time.

For instance, if I enter the command

```
C>dir
```

and then realize that I'm viewing the wrong directory, I can press [Ctrl][Break] to cancel the command. As soon as I press [Ctrl][Break], DOS will display the characters ^*C* on the screen and will return me to the DOS prompt, like this:

```
C>dir

 Volume in drive C has no label
 Directory of  C:\

AUTOEXEC BAT       290    3-06-90    2:59p
CONFIG   SYS       142    3-19-90    5:35p
COMMAND  COM     25308    7-18-88   12:00a
123            <DIR>      4-02-90   11:28a
WORD           <DIR>      4-28^C

C>_
```

Some new DOS users might want to be aware of this very helpful tool.

Robert Johnson
Atlanta, Georgia

A Mr. Johnson is right—[Ctrl][Break] comes in handy in a variety of situations. In fact, we use it regularly when we're writing a new batch file, and that file doesn't run perfectly the first time. When the batch file gets "hung up" in an endless loop, the easiest way to get out of the situation is to press [Ctrl][Break].

By the way, pressing [Ctrl]c has the same effect as pressing [Ctrl][Break]. You can use either of these key combinations to cancel the current command.

Clearing the screen

Q Although I've been using a PC for a couple of years, I recently discovered a simple command that I'm sure I will use over and over again: the CLS (or Clear Screen) command. Whenever I issue the command

```
C>cls
```

DOS immediately clears the screen and moves the DOS prompt to the top of the screen.

Continued on the back page

Contributing Editor Van Wolverton is the author of the bestselling books Running MS-DOS *and* Supercharging MS-DOS.

An overview of CONFIG.SYS

As I mentioned in my July column, DOS requires only three files in the root directory: COMMAND.COM, AUTOEXEC.BAT, and CONFIG.SYS. COMMAND.COM is the DOS *command interpreter*, the program that carries out the commands you type at the DOS prompt. AUTOEXEC.BAT, which I described in that column, contains DOS commands that are carried out each time you start or restart the system.

CONFIG.SYS, like AUTOEXEC.BAT, contains commands that DOS carries out each time you start the system. But these aren't DOS commands, they're a special set of commands called *configuration commands* that tell DOS how to manage its memory, keyboard, display, and other devices attached to your system. DOS carries out these commands before it carries out the commands in AUTOEXEC.BAT.

You probably have a CONFIG.SYS file

There's probably a file named CONFIG.SYS on your system, even if you didn't create it. It could have been installed at the computer store where you bought your computer, for example, or put there by someone else at your company who sets up new systems. Even if you set up your system yourself and didn't create a CONFIG.SYS file, some installation programs automatically create such a file (or modify the existing one) when you install a new program or device.

You can see whether you have a CONFIG.SYS file—and, if so, what's in it—by typing:

```
C>type  c:\config.sys
```

If DOS responds *File not found*, your system doesn't have a CONFIG.SYS file. But most likely you'll see something like this (the commands and order may vary):

```
break on
buffers=20
files=20
device=c:\ansi.sys
shell=c:\command.com  /e=512  /p
```

Don't worry if these commands look unfamiliar. Unlike the more common DOS commands, such as DIR or COPY, you don't use configuration commands every day, every week, or even every month. But when you install a new program or device, or want to tune up your system, it helps to understand the purpose of the commands and know how to use them.

Configuration commands

There isn't room here to describe the form and use of every configuration command, but the following brief descriptions should tell you whether you need the command in your CONFIG.SYS file:

- BREAK controls how often DOS lets you stop a command or program by pressing [Ctrl][Break] (or [Ctrl]-c).
- BUFFERS specifies the amount of memory DOS sets aside to handle disk files. Up to a certain point, the higher the number, the faster DOS can read and write a file.
- DRIVPARM tells DOS the operating characteristics (such as number of tracks and sectors per track) of a non-standard disk drive.
- FILES (and, in earlier versions of DOS, FCBS) specifies how many files DOS can use at one time. Certain programs, such as database programs, may need to use 20 or more files.
- INSTALL loads four DOS commands that remain in memory—Fastopen, Keyboard, Share, and Nlsfunc.
- LASTDRIVE specifies the highest drive letter that DOS will recognize. You may need this command if you use several disk drives—including a disk drive simulated in memory using VDISK.SYS or RAMDISK.SYS—or a large fixed disk partitioned as several volumes. You may also need a LASTDRIVE command if you use the SUBST command, which is explained on page 6.
- SHELL tells DOS where to find the command interpreter (usually COMMAND.COM). It also lets you increase the size of the *environment*, an area of memory that DOS uses.
- STACKS lets you control how much memory DOS reserves for its own temporary use.
- SWITCHES lets you prevent DOS from recognizing the additional keys (such as [F11] and [F12]) on the enhanced keyboard; some programs won't run properly if these keys are pressed unless you use this configuration command.

For a complete description of the form of these configuration commands, and more information about the circumstances under which you need them, see the DOS manual or one of the many DOS books available.

In addition to these commands, which control specific functions, CONFIG.SYS can contain one or more Device commands.

Device drivers

Many devices that you attach to your system, such as a scanner or CD-ROM drive, require a special program that tells DOS how to control (or *drive*) the device. You tell DOS the name and location of this program by including a Device command in CONFIG.SYS that specifies the programs' path and file names.

In virtually all cases, the program comes with the device; in most cases, an installation program is included, too, that automatically adds the necessary command to CONFIG.SYS (and, if necessary, one or more commands to AUTOEXEC.BAT). This can be a mixed blessing, though, because some programs don't modify CONFIG.SYS and AUTOEXEC.BAT intelligently, causing other devices—and in a few instances your entire system—to stop working properly. Some older installation programs, for example, simply created a new CONFIG.SYS or AUTOEXEC.BAT, wiping out your existing one; happily, this hardly ever happens now.

In addition to these drivers that come with nonstandard devices, DOS itself includes the following device drivers:

- ANSI.SYS lets programs control the color and cursor location of the display, and the assignment of special meanings to keys. To operate properly, many programs require ANSI.SYS.
- COUNTRY.SYS controls how DOS deals with different languages, including such characteristics as the keyboard layout and how the time and date are formatted.
- DISPLAY.SYS controls the character set used by the display for different languages.
- KEYBOARD.SYS defines the keyboard layout for different languages.
- MOUSE.SYS controls the operation of a mouse.
- PRINTER.SYS defines the character set used by several IBM printers to support different languages.
- RAMDRIVE.SYS or VDISK.SYS defines simulated fixed disk drives in memory.
- SMARTDRV.SYS sets aside some memory for use as a disk cache.
- XMAEM.SYS and XMA2EMS.SYS let DOS use extended memory as expanded memory.

The general form of the Device command is

```
DEVICE=filename
```

where *filename* is the path and filename of the device driver program. (Some device drivers, such as ANSI.SYS, require additional parameters of their own.)

Creating a CONFIG.SYS file

If your system doesn't have a CONFIG.SYS file, you should probably create one that includes at least the following configuration commands

```
break on
buffers=20
files=20
shell=C:\command.com /e=512 /p
```

Depending on the size and speed of your fixed disk and the type of programs you use, you may want to increase the numbers of both Files and Buffers. You can create or modify CONFIG.SYS with a text editor (like Edlin) or word processor that lets you store files with no formatting codes (often called *text-only* or ASCII files). Even if you already have a CONFIG.SYS file, you might want to check its contents and make sure that it includes at least the four commands listed above.

What if memory fails you?

Each configuration command sets aside memory for a particular use, reducing the amount of memory available for DOS and application programs. Terminate-and-stay-resident programs (TSRs, such as SideKick) consume even more memory. Where 640K once seemed limitless, it's often not enough to accommodate comfortably your CONFIG.SYS commands as well as your applications. When memory becomes tight, application programs will either run very slowly, or will not run at all.

If this happens to your system, you don't have many choices to free memory: You can change or eliminate some configuration commands, but this can similarly reduce the performance of your system; or you can eliminate some TSRs, but you probably paid good money for them and have become accustomed to their convenience.

Rather than making a permanent change to your system's configuration, there's a third alternative: You can create two or more alternate CONFIG.SYS files that define different configurations with different memory requirements, then write a batch file that selects the configuration that best matches the work you're planning to do and restarts your system. For a detailed look at configuring your computer see "Setting Up Multiple Configurations for Your Computer" in last month's *Inside Dos,* and "Enhancing the Dual Configuration Technique" which begins on page 8 of this issue.

Tailoring CONFIG.SYS to your hardware and software requires more understanding of how the computer works than does using a word processor or spreadsheet program. Unfortunately, this will be a fact of life as you add more hardware and software and until we get smarter installation programs.

Shortening directory paths using the SUBST command

If your hard disk is organized into several levels of directories, you probably find yourself typing very long path names as you issue commands from the DOS prompt or as you work with data files within your applications. For example, if you store your documents in the directory

C:\APPS\WP\LETTERS

you'll probably end up typing in that long path name whenever you need to access any of the files in that directory.

Fortunately, DOS provides a tool that helps eliminate the hassle of typing long path names: the SUBST command. In this article, we'll introduce you to the SUBST command, and we'll demonstrate a couple of situations where it can really save you keystrokes.

The SUBST command

The SUBST command lets you assign a drive letter to any directory. Once you've used the SUBST command to assign a drive letter to a directory, you can treat that directory as if it were a separate drive.

The basic form of the SUBST command is

```
subst drive path
```

where *drive* is the drive letter (followed by a colon) you want to assign, and *path* is the full path name of the directory you want to refer to with *drive*. Once you've used a SUBST command to assign a *drive* to a *path*, you can use the command

```
subst drive /d
```

to cancel the assignment, where *drive* is the drive letter whose assignment you want to cancel.

An example

For example, suppose you often refer to the files stored in the directory C:\APPS\WP\LETTERS, and that you want to treat that directory as if it were a separate drive E. To do this, simply issue the command

```
C>subst e: c:\apps\wp\letters
```

Once you've issued this command, you can work with the files in C:\APPS\WP\LETTERS much more easily than you could before. For instance, to display a listing of all the files stored in that directory, you can simply use the command

```
C>dir e:
```

instead of the command

```
C>dir c:\apps\wp\letters
```

Similarly, if you want to copy the file GRISSOM.DOC from the directory C:\APPS\WP\LETTERS to drive A, you can use the command

```
C>copy e:grissom.doc a:
```

instead of the command

```
C>copy c:\apps\wp\letters\grissom.doc a:
```

If you later decide you want to cancel the assignment of the letter E to the directory C:\APPS\WP\LETTERS, you'll need to issue the command

```
C>subst e: /d
```

Using valid drive parameters

Now that you know how the SUBST command works, you need to be aware of a few rules governing the *drive* parameter of this command. First of all, make sure you don't use a letter assigned to a physical drive as the *drive* parameter. If you do, DOS will not allow you to access that physical drive until you use another SUBST command to cancel the assignment of that drive letter.

By default, the highest drive letter DOS will recognize is E. If you need to use drive letters higher than E, you'll need to insert a LASTDRIVE command into your CONFIG.SYS file that takes the form

```
lastdrive=letter
```

where *letter* (without the colon) is the highest drive letter you want to use. As you might guess, you can specify any letter up to Z for your *letter* parameter. If you attempt to use a drive letter in a SUBST command that is higher than the letter specified in CONFIG.SYS, DOS will display the message *Invalid parameter*. (For a detailed discussion of the CONFIG.SYS file, read the article that begins on page 4).

For example, if you want to use drive letters up to Z in your SUBST commands, you would add to your CONFIG.SYS file the command

```
lastdrive=z
```

Listing the substitutions

You can view the drive substitutions you've made whenever you want by simply typing the command *subst* without any parameters. When you press ↵, DOS will display a list of all the drive assignments you've made. For example, if you've previously made drive substitutions using the commands

```
C>subst  e:  c:\apps\wp\letters
C>subst  f:  c:\apps\wp\forecast
```

you can verify that DOS made these substitutions by entering the command

```
C>subst
```

When you press ↵, DOS will respond

```
E:  =>  C:\APPS\WP\LETTERS
F:  =>  C:\APPS\WP\FORECAST
```

Shortening your path

In addition to saving you keystrokes at the DOS prompt, the SUBST command might allow you to reduce the length of your system path. If you've installed applications in some relatively "deep" directories on your hard disk, you can precede the PATH command in your AUTOEXEC.BAT file with a SUBST command that assigns drive letters to those application directories. Then, you can use the new drive letter in your PATH command instead of using the longer directory names.

For example, suppose you've installed several applications in directories off of the C:\APPS directory, as shown in Table A.

Table A

You've installed...	in the directory...
Lotus 1-2-3	C:\APPS\123
WordPerfect	C:\APPS\WP
Harvard Graphics	C:\APPS\HG

Now, if you want to include in your system path every application directory listed in Table A as well as your DOS directory (C:\DOS), you'll want to include the following command in AUTOEXEC.BAT:

```
path  c:\dos;c:\apps\123;c:\apps\wp;c:\apps\hg
```

To shorten this path, you could first use the following SUBST command to assign the drive letter *K* to the directory C:\APPS:

```
subst  k:  c:\apps
```

Then, you can use the drive letter *K* in place of *C:\APPS* everywhere in the PATH command, like this:

```
path  c:\dos;k:\123;k:\wp;k:\hg
```

For this simple example, the SUBST command reduces the length of the path from 40 to 25 characters. Of course, the savings will vary according to both the number of directories you include in the path, and the length of the path name to which you assign a drive letter.

One drawback

The only drawback to using SUBST commands to assign drive letters to directories is that each substitution occupies a few bytes of memory. For this reason, make sure you don't go overboard with SUBST commands—use them to make only the substitutions you'll use on a regular basis.

Conclusion

If you've set up multiple levels of directories on your hard disk, you'll often find yourself typing in long path names as you access files from DOS and from within applications. In this article, we've shown you how to use the SUBST command to eliminate the hassle of typing long path names.

Enhancing the dual-configuration technique

The idea for this article was submitted by Contributing Editor Tim Landgrave. Tim is the Director of The Cobb Group's Developer Group.

In last month's issue, we showed you a couple of batch files that make it easy to set up two configurations for your computer. This month, we'll show you how to enhance that technique using the soliciting method we explain in this month's lead article. Even if you don't need to set up dual configurations for your machine, you'll probably benefit from reading this article, since it covers several useful batch file programming techniques.

A brief review

Suppose you want to configure your computer as either a stand-alone machine or as a network station. Last month, we showed you how to do this by first creating two sets of AUTOEXEC.BAT and CONFIG.SYS files, and saving those sets of files in the directory C:\SYS under the names

```
AUTOEXEC.NET
CONFIG.NET
AUTOEXEC.STD
CONFIG.STD
```

Once you've placed these files in the C:\SYS directory, you can use the batch files we showed you last month to configure your machine the way you want. As you may recall, the batch file that configures your machine as a network station (which we named STARTNET.BAT), replaces the current AUTOEXEC.BAT and CONFIG.SYS files in your root directory with the files

```
AUTOEXEC.NET
CONFIG.NET
```

and uses the utility REBOOT.COM to reboot your machine. Similarly, the batch file that configures your machine as a stand-alone machine (which we named NONET.BAT), replaces the current AUTOEXEC.BAT and CONFIG.SYS files in your root directory with the files

```
AUTOEXEC.STD
CONFIG.STD
```

and then uses REBOOT.COM to reboot your machine.

The new technique

Let's expand upon last month's technique by creating a single AUTOEXEC.BAT file that lets you choose a particular configuration when you reboot. Figure A shows the basic form of an AUTOEXEC.BAT file that will do the trick.

Figure A

```
@echo off
cls
prompt $p$g
path  c:\dos;c:\util;c:\batch;c:\123
if exist c:\config.std goto net

:STDALONE
echo Your machine is currently configured
   as a STAND ALONE MACHINE
echo Press Y to continue, or press N to
   reboot as a network station
keypress
if errorlevel 3 goto stdcont
ren c:\config.sys config.std
ren c:\config.net config.sys
reboot

:NET
echo Your machine is currently configured
   as a NETWORK STATION
echo Press Y to continue, or press N to
   reboot as a stand alone machine
keypress
if errorlevel 3 goto netcont
ren c:\config.sys config.net
ren c:\config.std config.sys
reboot

:NETCONT
echo Network startup commands go here

:STDCONT
echo Stand alone startup commands go here

:END
```

This AUTOEXEC.BAT file allows you to choose a particular configuration when you reboot.

Don't be overwhelmed by the size of the batch file in Figure A—it's pretty easy to follow once you're aware of what's going on. Before we explain the batch file line by line, however, it's important for you to understand the general technique that it uses to configure your system.

How it works

The root directory of your boot disk should always contain two CONFIG files—one for booting up on the network, and one for booting up as a stand-alone machine. However, instead of naming the two files

```
CONFIG.NET
CONFIG.STD
```

you'll name one of them CONFIG.SYS (the one you want to use for your original configuration). When you reboot your machine, it will configure itself with the settings in CONFIG.SYS, then run AUTOEXEC.BAT.

The first thing our AUTOEXEC.BAT file will do is look at the CONFIG files in the root directory to determine whether the machine is currently configured as a network station or as a stand-alone machine. Then, it will ask if you want to continue booting with the selected configuration. If you choose to continue with the current configuration, AUTOEXEC.BAT will tell DOS to execute the batch instructions associated with that particular configuration.

However, if you tell AUTOEXEC.BAT that you want to switch configurations, it will rename the two CONFIG files in the root directory and reboot. For instance, suppose your boot disk contains the files CONFIG.SYS (which contains your stand alone settings), and CONFIG.NET. When you tell AUTOEXEC.BAT to switch configurations, it will change the name of CONFIG.SYS to CONFIG.STD, and will change the name of CONFIG.NET to CONFIG.SYS:

```
CONFIG.SYS  ➡  CONFIG.STD
CONFIG.NET  ➡  CONFIG.SYS
```

Now that the appropriate configuration files are in place, AUTOEXEC.BAT will reboot the machine. The next time you reboot and tell AUTOEXEC.BAT to switch configurations, it will rename the files before it reboots:

```
CONFIG.SYS  ➡  CONFIG.NET
CONFIG.STD  ➡  CONFIG.SYS
```

Breaking down the batch file line by line

Now that you have a general grasp of how the whole process works, let's look at the batch file in Figure A line by line. As usual, the batch file first issues the command *@echo off*, which tells DOS to suppress the display of the batch file commands as it executes them. The next two commands in the batch file

```
prompt $p$g
path  c:\dos;c:\util;c:\batch
```

define the desired prompt and path.
Next, AUTOEXEC.BAT checks to see which configuration was specified at startup using the command

```
if exist c:\config.std goto net
```

This command checks to see if the file C:\CONFIG.STD exists. If it does, then AUTOEXEC.BAT knows the file CONFIG.SYS contains network settings, and that the computer is currently configured as a network station. At that point, it will go to the section of the batch file labelled NET and execute the instructions there. However, if the file C:\CONFIG.STD does not exist, AUTOEXEC.BAT knows that CONFIG.SYS contains the stand-alone settings, and that the computer is currently configured as a stand-alone machine. Consequently, DOS will not execute the *goto net* portion of the IF command, and will allow execution to continue with the STDALONE section of the batch file.

As you can see in Figure A, the STDALONE and NET sections of the batch file are very similar. We'll explain the commands in the STDALONE section first, then point out the differences between it and the NET section.

The first two commands in the STDALONE section

```
echo Your machine is currently configured
   as a STAND ALONE MACHINE
echo Press Y to continue, or press N to
   reboot as a network station
```

tell you that the current configuration is for a stand-alone machine, and ask if you want to continue booting under the current configuration. The next command

```
keypress
```

executes the program KEYPRESS.COM, which waits for you to press Y or N. As soon as you press one of these keys, DOS will execute the next command

```
if errorlevel 3 goto stdcont
```

which checks to see if you pressed Y. If you did, meaning that you want to continue booting as a stand-alone machine, DOS will branch down to the section of the batch file labelled STDCONT, which contains the startup commands for a stand-alone workstation. If you didn't press Y, however, DOS will not branch to STDCONT and will instead execute the next two commands in the batch file:

```
ren  c:\config.sys  config.std
ren  c:\config.net  config.sys
```

These two commands rename the CONFIG files in the root directory according to the scheme we described earlier; the current CONFIG.SYS becomes CONFIG.STD, and CONFIG.NET becomes the new CONFIG.SYS. Finally, the last command in the STDALONE section of the batch file

```
reboot
```

executes the REBOOT.COM program, which reboots the machine, and activates the new CONFIG.SYS settings.

(In case you missed last month's journal you can use the Debug scipt in Figure B to execute the file REBOOT.COM.)

Figure B

```
n reboot.com
e 100 b8 40 00 8e d8
e 105 b8 34 12 bb 72
e 10a 00 89 07 ea 00
e 10f 00 ff ff
rcx
18
w
q
```

Use this Debug script to create the utility REBOOT.COM.

As we mentioned, the NET section of the AUTOEXEC-.BAT file in Figure A is similar to the STDALONE section. As you'd expect, the commands in this section first display the current configuration and ask you if you want to change configurations. If you press Y, meaning that you want to continue booting under the current configuration, DOS branches down to the NETCONT section of the batch file and executes the commands there. However, if you press N, meaning that you want to change configurations, DOS renames CONFIG.SYS CONFIG.NET, renames CONFIG.STD CONFIG.SYS, and reboots the machine to activate the new configuration.

The rest of the commands in the batch file are pretty self-explanatory. The NETCONT section of the batch file contains the commands you want DOS to execute when you boot up as a network station. This section should contain all the commands you'd normally include in AUTOEXEC.BAT for a network station. The last command in this section

```
goto end
```

tells DOS to skip over the STDCONT section and to go to the label END at the very end of the batch file.

Similarly, the STDCONT section of the batch file simply contains the commands you want DOS to execute for a stand-alone machine. You'll want to include in this section all the commands you'd normally include in AUTOEXEC.BAT for a stand-alone machine. When DOS reaches the label END, it will cease execution of AUTOEXEC-.BAT and will display a DOS prompt.

Soliciting input in a batch file

Continued from page 1

next to the prompt for line 12 and will return you to the Edlin prompt:

```
       12:*^C
*_
```

Finally, type the letter *e* and press ↵ to end the current editing session and return to DOS:

```
*e
C>_
```

Now that you've created a script of Debug instructions called KEYPRESS.SCR, you're ready to run Debug and create the file KEYPRESS.COM. To do this, simply issue the command

```
C>debug < keypress.scr
```

Immediately, Debug will execute the instructions in the file KEYPRESS.SCR, and will deposit the file KEYPRESS.COM into the current directory.

Finally, be sure to copy the file KEYPRESS.COM into a directory whose name is on the path. That way, DOS will always be able to find and execute the file KEYPRESS.COM.

Using KEYPRESS.COM in a batch file

Once you've created KEYPRESS.COM, it's easy to solicit user input. You first enter an ECHO command into your batch file that asks the user to press either Y, N, or ESC. Then, you execute the utility KEYPRESS.COM to tell DOS to wait and see which is pressed. If you press any key other than Y, N, or ESC, the computer will beep and continue waiting for a valid response. However, if you press Y, N, or ESC, DOS will set the value of the variable *errorlevel* to either 3, 2, or 1, respectively. The batch file can then determine which key was pressed using a series of IF statements that 1) check the value of *errorlevel*, and 2) branch to the appropriate part of the batch file. Let's consider an example. (By the way, KEYPRESS.COM is *not* sensitive to capitalization—it looks for *y*, *Y*, *n*, *N*, or the [Esc] key.)

Figure B shows a sample batch file named KEYTEST.BAT that prompts for Y, N or ESC, then branches to the appropriate part of the batch file.

Figure B

```
@echo off
echo Press Y, N, or ESC
keypress
if errorlevel 3 goto yes
if errorlevel 2 goto no
if errorlevel 1 goto esc
:YES
echo You selected YES
goto end
:NO
echo You selected NO
goto end
:ESC
echo You selected ESC
:END
```

You can use KEYTEST.BAT to test your copy of KEYPRESS.COM.

Although KEYTEST.BAT might look pretty complex at first, it's actually very simple. Let's break down this batch file line by line and see how it works. Before we begin, however, notice that KEYTEST.BAT contains four lines that begin with a colon and that are followed by a single word:

```
:YES
:NO
:ESC
:END
```

These lines don't instruct DOS to do anything—they are simply labels that mark an important place in the batch file. As you'll see in a moment, you can use a GOTO command to tell DOS to go to a label and to carry out the commands that follow the line on which that label appears.

Now let's analyze KEYTEST.BAT step by step. The first command

```
@echo off
```

simply tells DOS not to display the batch file's commands as it executes them. By the way, if you're using version 3.2 or earlier of DOS, you'll need to omit the @ sign in front of the *echo off* command. (As we mentioned last month, the @ sign hides the *echo off* command as well.)

The next command in KEYTEST.BAT

```
echo Press Y, N, or ESC
```

tells DOS to display on the screen the message *Press Y, N, or ESC*. Next, DOS will execute the command

```
keypress
```

which executes the program KEYPRESS.COM. At this point, DOS will wait for you to press either Y, N, or ESC. If you press any key other than these three, DOS will simply beep and continue waiting for a valid response.

As soon as you press either Y, N, or ESC, DOS will set the value of *errorlevel* to either 3, 2, or 1, depending on which key you pressed, and will continue executing the commands in the batch file.

The next command in KEYTEST.BAT

```
if errorlevel 3 goto yes
```

checks to see if *errorlevel* is equal to 3. If *errorlevel* does not equal 3, then DOS immediately executes the next command in the batch file; however, if *errorlevel* equals 3, then DOS carries out the command *goto yes*, which tells DOS to carry out the command that immediately follows the line containing the label :YES.

As we just mentioned, if *errorlevel* is not equal to 3, DOS will execute the following command in KEYTEST.BAT:

```
if errorlevel 2 goto no
```

Of course, this command checks to see if *errorlevel* is equal to 2. If it isn't, DOS will carry out the next command in the batch file. If *errorlevel* is equal to 2, however, DOS will carry out the command *goto no*, which tells DOS to execute the command that follows the label :NO.

If *errorlevel* is not equal to either 3 or 2, DOS will execute the last IF command in KEYTEST.BAT:

```
if errorlevel 1 goto esc
```

This command checks to see if *errorlevel* is equal to 1. If it is, DOS will branch to the command that follows the label :ESC if it isn't, DOS will execute the next command in the batch file.

Because the first two IF commands in this particular batch file check to see if *errorlevel* is equal to 3 or 2, DOS will make it to the third IF command *only* when *errorlevel* is equal to 1; therefore, we really don't need to use the third IF command to test the value of *errorlevel*. However, we went ahead and included this extra IF command to illustrate clearly the mechanics of branching.

After DOS makes it past the IF statements in KEYTEST.BAT, it will branch to either the YES, NO, or ESC section of the batch file. As we explained earlier, the lines containing the labels :YES, :NO, and :ESC mark these three sections. Don't forget to precede the labels in your batch files with a colon—if you forget, DOS will not recognize them as labels, and will assume that they're commands.

As you can see, the YES section of KEYTEST.BAT contains the commands

THE COBB GROUP
Postmaster:
Monthly Technical Journal
Please Rush!
Second Class Postage

Please include account number from label with any correspondence

```
echo You selected YES
goto end
```

As you know, the command *echo You selected YES* tells DOS to display the message *You selected YES*. Of course, the command *goto end* tells DOS to branch to the command immediately following the label :END, which appears on the last line of the batch file. Because there aren't any lines following the label :END, DOS will stop executing the batch file at this point and redisplay the DOS prompt.

The NO and ESC sections of KEYTEST.BAT are similar to the YES section. The commands in these sections simply tell DOS to display an appropriate message on the screen and to branch to the label :END, which ends the execution of this batch file.

Testing the batch file

By the way, KEYPRESS.COM requires that you test for the Y key first, then for the N key, and finally for the ESC key, in that order. If you test for the ESC key first using the command

```
if errorlevel 1 goto esc
```

DOS will always issue the *goto esc* command. For this reason, make sure you don't test for the ESC key first, and that you test for Y before you test for N. Now that we've explained how the batch file in Figure B works, let's test it. Assuming you've named the batch file KEYTEST.BAT, you can run the batch file by typing

```
C>keytest
```

Immediately, DOS will respond with the prompt

```
Press Y, N, or ESC
_
```

If, at this point, you press either y or Y, DOS will display the message

```
You selected YES
```

and will redisplay the DOS prompt. If you run KEYTEST.BAT again and press either n or N, DOS will display the message *You selected NO* and return you to the DOS prompt. Finally, running KEYTEST.BAT and pressing the ESC key will tell DOS to display the message *You selected ESC*.

Conclusion

In this article, we've shown you how to solicit user input from within a batch file. We first showed you how to use Debug to create the program KEYPRESS.COM, then we demonstrated how to take advantage of this program in a batch file. Now that you're familiar with this technique, you can use it to solicit input in your own batch files. For a closer look at this technique in action, read the article that begins on page 8.

Letters

Continued from page 3

Although CLS doesn't really do that much, I use it as often as I'm working at the DOS prompt and cluttering my screen.

Sheila Pressey
Mount Vernon, New York

A Ms. Pressey's letter reminds us that sometimes the most simple commands are also the most useful. Many of our readers have mentioned that they use the CLS command in nearly all their batch files, as well as at the DOS prompt. We've found that the CLS command comes in handy after we've used someone else's computer and we don't want to leave their screen cluttered with the commands we've issued. As soon as we finish working with their computer, we issue the CLS command, and the user can return to a nice, clean display.